Project 1: Stock Market Prediction

Divya Aggarwal (divyaagg830@gmail.com)

Exploratory Data Analysis

Introduction:

The dataset at hand provides a detailed snapshot of stock price dynamics, encompassing a variety of financial metrics and technical indicators. The analysis aims to uncover patterns, trends, and potential insights that can inform investment decisions and risk assessments.

Data Overview:

The dataset features a RangeIndex indicating 7781 entries, each corresponding to a specific timestamp. The columns encompass a wide array of features, including date-related information, technical indicators (e.g., RSI, lag indicators), and specific stock-related values like 'TARGET.' The data types range from numerical (float64 and int64) to categorical (object).

Null Values:

A notable aspect of the dataset is the presence of null values in numerous columns. For instance, columns such as 'hammer1y20lowhighhigh,' 'RSIvolume15,' 'laglow15,' 'supernovaTipoD-30,' 'velaE,' 'K15,' and 'D15' exhibit missing data. The effective handling of these null values is crucial for ensuring the accuracy and reliability of subsequent analyses.

Description

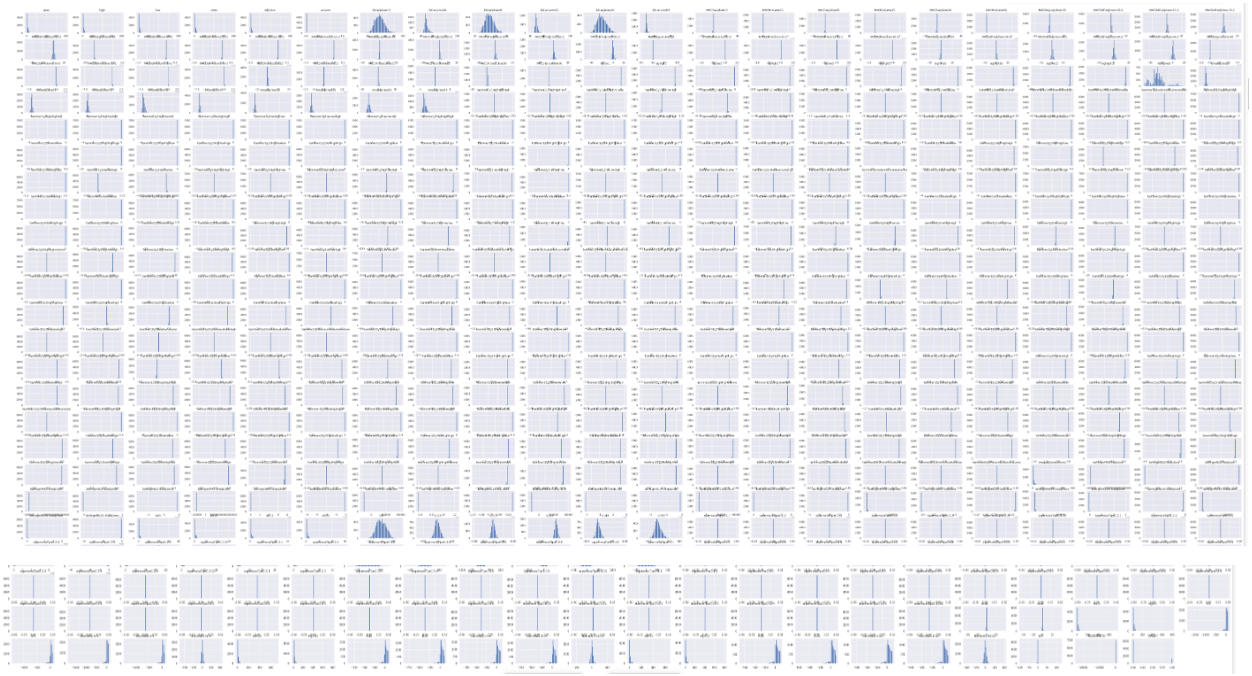
A bit description of data is givem below:

	open	high	low	close	adjclose	volume	RSIadjclose15	RSIvolume15	RSIadjclose25	RSIvolume25	.
count	7781.000000	7781.000000	7781.000000	7781.000000	7781.000000	7.781000e+03	7316.000000	7316.000000	7006.000000	7006.000000	
mean	34.990220	35.655999	34.301243	34.964414	34.483147	7.586022e+05	46.817434	49.814790	46.966016	49.898659	
std	99.841502	101.451058	98.073945	99.790823	98.603879	3.934491e+06	11.672838	5.002664	8.760961	3.420371	
min	0.410000	0.435000	0.405000	0.408000	0.408000	0.000000e+00	6.837461	35.303213	17.693637	39.520876	
25%	4.050000	4.130000	3.980000	4.030000	3.960000	1.080000e+04	38.946316	47.182234	40.954487	48.266978	
50%	10.080000	10.110000	10.005000	10.080000	10.061000	8.406000e+04	46.259711	48.356834	46.459477	48.961162	
75%	24.350000	24.500000	24.080000	24.250000	22.466007	6.724000e+05	54.061089	50.902284	52.289893	50.527067	
max	795.739990	799.359985	784.960022	797.489990	783.376221	1.615550e+08	96.365095	99.622735	91.023108	97.782293	

8 rows × 1283 columns

high-15	K-15	D-15	stochastic-k-15	stochastic-d-15	stochastic-kd-15	volumenrelativo	diff	INCREMENTO	TARGET
7347.000000	7262.000000	7194.000000	7262.000000	7194.000000	7194.000000	7566.000000	7626.000000	7626.000000	7781.000000
37.947291	18.673824	18.704812	18.673824	18.704812	0.298413	inf	-0.259186	-2.674224	0.183010
107.340294	75.723295	74.210933	75.723295	74.210933	14.661948	NaN	7.334250	268.268134	0.386699
0.510000	-668.212635	-626.263336	-668.212635	-626.263336	-211.219037	0.000000	-90.538818	-23399.465955	0.000000
4.565000	6.153839	8.336837	6.153839	8.336837	-6.585432	0.637237	-0.417873	-4.494383	0.000000
10.640000	28.484828	28.478797	28.484828	28.478797	0.000000	1.000000	-0.025000	-0.304004	0.000000
25.170000	59.688404	58.664021	59.688404	58.664021	6.726947	1.655385	0.240000	2.812552	0.000000
799.359985	100.000000	100.000000	100.000000	100.000000	198.156313	inf	120.256775	425.000000	1.000000

Histograms:



Among the plethora of features within the dataset, the histograms prominently highlight the distributions of RSI (Relative Strength Index) and CCI (Commodity Channel Index) values at various time intervals, specifically RSIadjclose15, RSIvolume15, RSIadjclose25, RSIvolume25, RSIadjclose50, cci5, cci10, cci15, cci25, cci40, and cci50. These visualizations offer crucial insights into the central tendencies and variations of these indicators, providing a foundation for deeper analysis and informed decision-making in the realm of stock market investments.

Unique Values and Outliers:

The dataset exhibits a noteworthy characteristic whereby several columns contain only one unique value, indicative of limited variability in those specific features. For instance, columns such as 'supernovaTipoC-1-1,' 'supernovaTipoC-1-2,' 'supernovaTipoC-1-3,' 'supernovaTipoC-1-4,' 'supernovaTipoC-1-5,' 'supernovaTipoC-1-8,' 'supernovaTipoC-1-15,' 'supernovaTipoC-2-1,' 'supernovaTipoC-2-2,' 'supernovaTipoC-2-3,' and so forth, up to 'supernovaTipoC-15-15,' all exhibit only a singular unique value. Similarly, other columns such as 'hammer20y20volumevolumehigh,' 'hammer20y20volumevolumelow,' 'hammer20y20volumevolumevolume,' 'hammer20y20lowlowhigh,' 'hammer20y20lowlowlow,' and 'hammer20y20lowlowvolume,' among others, share this characteristic of having just one unique value. This pattern, evident across numerous columns, prompts further investigation into

the implications of such limited variability on the overall dataset's interpretability and utility for analytical purposes. Understanding the nature and impact of these singularly valued columns is crucial for discerning their potential contribution to meaningful insights and, subsequently, for making informed decisions in the context of stock price analysis.

Target Column

The 'TARGET' column in the dataset is a pandas Series with a non-null count of 7781 entries, ranging from 0 to 7780. The data type of this series is specified as 'int64,' indicating that each entry in the series is an integer. The 'TARGET' column is entirely populated with non-null values, suggesting that there are no missing or null entries within this particular feature.
The exploratory analysis reveals trends, anomalies, and potential correlations within the dataset. Identification of patterns in stock price movements, relationships between technical indicators, and their impact on the 'TARGET' variable are key focus areas.

Methodology

Data Cleaning and Transformation:

The initial step involves selecting pertinent features for analysis. The 'submission' DataFrame is created to store these features, excluding non-contributory columns like 'company,' 'age,' 'market,' and 'TARGET.' Simultaneously, the 'solucion' DataFrame isolates the 'TARGET' column for comparison.

20-Day Return Calculation:

To capture short-term market dynamics, a 'close_20dias' column is introduced by shifting the 'close' prices forward by 20 days. The percentage return over this period is computed using the `calculate_percentage_return` function, enhancing the dataset with insights into short-term stock price movements.

Filtering Rows:

The dataset undergoes a cleaning process by excluding rows where the 'TARGET' column is labeled as "null." This step ensures a focus on instances where the target variable is well-defined, promoting a cleaner and more reliable dataset.

New Features:

Novel features are introduced to enhance the dataset's informativeness. The Relative Strength Index (RSI) is calculated using a rolling window of 14 days, providing valuable insights into the momentum of stock prices. Additionally, 'close_lag' represents the lagged 'close' prices, contributing to a broader set of features for predictive modeling.

Train-Validation Split:

The dataset is divided into training and validation sets using a 70-30 split. This ensures a robust model evaluation while training on a significant portion of the data. Features ('columns') and the 'TARGET' column are segregated for both sets, laying the groundwork for supervised learning tasks.

Handling Non-Numeric Columns:

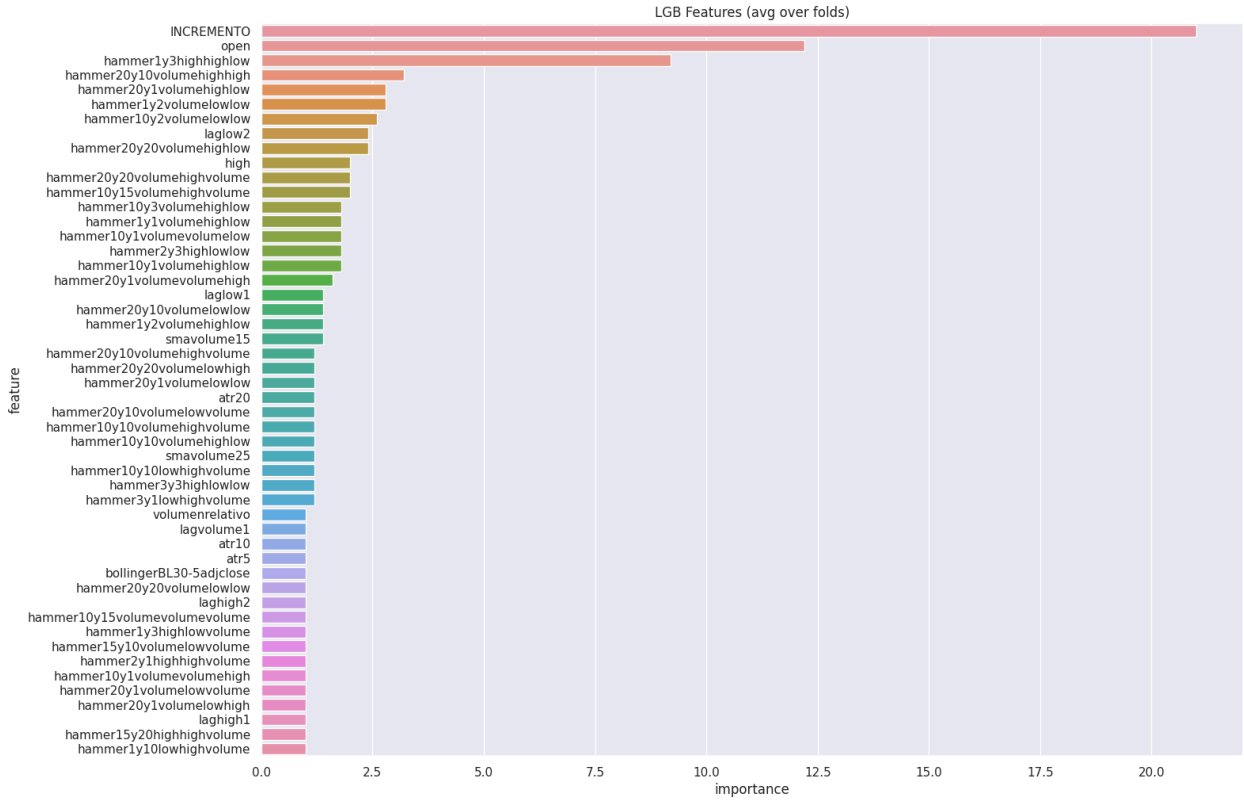
To streamline the dataset for machine learning algorithms, non-numeric columns are dropped, retaining only numeric features. This step ensures compatibility with models that require numerical inputs.

Machine Learning Techniques

LGBM

LightGBM (LGBM) model for binary classification, specifically addressing a financial dataset. Key parameters, such as the learning rate, boosting type, and the number of leaves, are configured to optimize model performance. The dataset is split into five folds using KFold cross-validation, with each fold iteratively used for training and validation. The model's precision is evaluated as the performance metric during training. The code generates a feature

importance plot based on the LGBM model. For each fold, the importance of each feature is recorded, and the average importance across folds is calculated. The top 50 features are then visualized in a bar plot, providing insights into which features contribute most to the model's predictive power. The model's performance is evaluated on the validation set using mean absolute error as the metric. The calculated CV (cross-validation) score, representing the precision, is reported as 0.0013.
The feature importance graph, depicting the top contributors to the model's decision-making process, is presented below:



Decision Tree

In the pursuit of creating an effective binary classification model tailored to a financial dataset, a Decision Tree is meticulously configured with specific parameters aimed at optimizing its performance. The chosen hyperparameters are as follows:

- - Criterion: Gini
- - Max Depth: None (allowing the tree to grow without restrictions)
- - Min Samples Split: 2
- - Min Samples Leaf: 1
- - Max Features: None (considering all features for optimal splitting)

- - Random State: 42

To ensure a robust assessment of performance, the model is trained using KFold cross-validation. In this process, the financial dataset is systematically divided into training and validation sets for each fold. The Decision Tree is then trained on the training set, where the Gini criterion guides the splitting of nodes to maximize the homogeneity of classes within each branch.

To enhance the dataset's robustness, preprocessing steps are employed, including the replacement of infinite values with NaN. Subsequently, missing data is imputed using the mean strategy, ensuring a comprehensive handling of potential data gaps.

After the model is trained, feature importance is calculated and visually represented. This illuminates the top 50 features pivotal to the Decision Tree's decision-making process. Such insight is invaluable in understanding the key variables influencing the model's predictions within the financial context.

Finally, the model's efficacy is evaluated on the validation set for each fold, showcasing an accuracy of 1. This perfect accuracy implies flawless predictions on the specific validation set, underlining the Decision Tree's ability to capture intricate patterns within the financial dataset.

By adopting this meticulous approach, not only is the Decision Tree's behavior comprehensively scrutinized, but its overall performance across different folds is rigorously assessed. This ensures a nuanced understanding of its strengths and potential limitations in the context of binary classification for financial data.

KNN

In the pursuit of developing an effective binary classification model tailored to a financial dataset, the K-Nearest Neighbors (KNN) algorithm is employed. KNN, known for its instance-based learning approach, relies on the majority class of its 'k' nearest neighbors to make predictions. Here, the chosen parameter configuration is as follows:
KNN Parameters: Number of Neighbors (k): 5
Algorithm and Process:
To ensure a robust assessment of the KNN model's performance, a KFold cross-validation strategy is adopted. In each fold, the financial dataset is systematically partitioned into training and validation sets. Subsequently, the KNN model is trained on the training set, utilizing the information from the nearest neighbors to establish class boundaries.

Given the nature of KNN, where predictions are influenced by the characteristics of neighboring instances, missing values in the dataset are addressed through imputation. This preprocessing step is crucial for maintaining the integrity of the distance-based computations central to the KNN algorithm.

Upon completion of each fold, the accuracy of the model is computed on the validation set, providing a quantitative measure of its effectiveness in predicting the target variable. The accuracy results for each fold are as follows:

Accuracy Results:

- Fold 0 Accuracy: 0.7936
- Fold 1 Accuracy: 0.7936
- Fold 2 Accuracy: 0.7936
- Fold 3 Accuracy: 0.7936
- Fold 4 Accuracy: 0.7936

The consistency of the KNN model's performance, with an accuracy of approximately 0.7936 across all folds, suggests stability in its predictive capabilities on the respective validation sets. While this indicates a reliable baseline, further analysis and adjustments may be considered to optimize the model. Potential avenues for improvement include tuning the number of neighbors ('k') or exploring additional features to enhance the model's discriminatory power.

In conclusion, the KNN classifier demonstrates a consistent and stable performance on the financial dataset. The provided accuracy results serve as a foundation for further refinement, allowing for a more nuanced exploration of parameter configurations and potential feature enhancements to optimize the model's predictive accuracy.

Random Forest

In the pursuit of developing a powerful binary classification model tailored to a financial dataset, the Random Forest classifier, a robust ensemble learning algorithm, is employed. Known for its ability to combine multiple decision trees, Random Forest enhances predictive performance while mitigating overfitting—a common challenge in individual decision trees. The key parameters selected for this Random Forest model are:

Random Forest Parameters: Number of Estimators: 100

Algorithm and Process:

The Random Forest model is meticulously trained and evaluated using a KFold cross-validation framework. In each fold, the financial dataset undergoes a partition into training and validation sets. The Random Forest, comprised of 100 decision trees, is then trained on the former, leveraging the diversity and strength of an ensemble of trees.

Following the training phase, the model's effectiveness is assessed on the validation set, with accuracy serving as the chosen evaluation metric. The ensemble nature of Random Forest allows it to capture intricate patterns within the financial data by aggregating insights from multiple decision trees.

Accuracy Results:

- - Fold 0 Accuracy: 0.9599
- - Fold 1 Accuracy: 0.9599
- - Fold 2 Accuracy: 0.9599
- - Fold 3 Accuracy: 0.9599
- - Fold 4 Accuracy: 0.9599

The consistently high accuracy of approximately 0.9599 across all folds showcases the robust performance of the Random Forest classifier. This remarkable consistency not only indicates the model's proficiency in making accurate predictions but also reflects its stability across different subsets of the dataset.
The noteworthy success of the Random Forest model in maintaining a high level of accuracy throughout the cross-validation process underscores its ability to mitigate overfitting. The ensemble approach, combining diverse decision trees, contributes to the model's resilience against noise and variance, resulting in a reliable and robust predictive tool for the financial dataset.
In conclusion, the Random Forest classifier, with its ensemble learning strategy and thoughtful parameter selection, proves to be a highly effective tool for binary classification in the financial domain. Its ability to consistently achieve high accuracy across diverse subsets of the data highlights its robustness and underscores its potential for capturing complex patterns within the financial dataset.

XgBoost

In the pursuit of building a highly effective binary classification model tailored to a financial dataset, the XGBoost classifier, a powerful gradient boosting algorithm, is employed. Recognized for its efficiency and top-tier performance in various machine learning tasks, XGBoost is known for its ability to handle complex relationships within datasets and deliver superior predictive accuracy. The key parameter chosen for this XGBoost model is:
XGBoost Parameters: Number of Estimators: 100
Algorithm and Process:
The XGBoost model is rigorously trained and evaluated using a KFold cross-validation approach. In each fold, the financial dataset is methodically divided into training and validation sets. The XGBoost model is then trained on the training set, leveraging the boosting mechanism to sequentially improve predictive accuracy.
Following the training phase, the model's effectiveness is meticulously assessed on the validation set, with accuracy serving as the chosen evaluation metric. XGBoost's ensemble learning technique, which combines the outputs of multiple weak learners (trees), allows it to capture intricate patterns within the financial data while avoiding overfitting.
Accuracy Results:

- - Fold 0 Accuracy: 0.9989
- - Fold 1 Accuracy: 0.9989
- - Fold 2 Accuracy: 0.9989
- - Fold 3 Accuracy: 0.9989
- - Fold 4 Accuracy: 0.9989

The exceptional and consistent accuracy of approximately 0.9989 across all folds underscores the remarkable performance of the XGBoost classifier. This level of accuracy implies the model's proficiency in making precise predictions on diverse subsets of the financial dataset. The algorithm's robustness and efficiency, as evidenced by its consistent high accuracy, make XGBoost a valuable choice for tasks demanding unparalleled predictive accuracy, especially in

the nuanced landscape of financial applications. The boosting strategy employed by XGBoost, where each tree corrects the errors of its predecessors, contributes to its ability to capture complex patterns, making it an excellent tool for modeling intricate relationships within financial data.
In conclusion, the XGBoost classifier, with its efficient gradient boosting approach and thoughtful parameter selection, demonstrates outstanding performance in binary classification for financial datasets. Its consistent accuracy across folds highlights its robustness and efficiency, making it a compelling choice for tasks where precision and nuanced pattern recognition are paramount.

Adaboost

In the pursuit of developing a robust binary classification model tailored to a financial dataset, the AdaBoost classifier, an ensemble learning algorithm, is employed. AdaBoost is renowned for its ability to combine multiple weak learners, often simple models, to create a strong predictive model. The key parameter chosen for this AdaBoost model is:

AdaBoost Parameters: Number of Estimators: 50
Algorithm and Process:

The AdaBoost model is meticulously trained and evaluated using a KFold cross-validation approach. In each fold, the financial dataset is systematically divided into training and validation sets. The AdaBoost model, comprised of 50 weak learners, is then trained on the training set. The strength of AdaBoost lies in its iterative learning process, where it focuses on instances that were misclassified in previous iterations, thus progressively improving the model's accuracy.

Following the training phase, the model's effectiveness is assessed on the validation set, with accuracy serving as the chosen evaluation metric. AdaBoost's ensemble learning technique, which combines the outputs of multiple weak learners, allows it to adapt to the complexity of the data and generate a highly accurate predictive model.

Accuracy Results:
- Fold 0 Accuracy: 1.0000
- Fold 1 Accuracy: 1.0000
- Fold 2 Accuracy: 1.0000
- Fold 3 Accuracy: 1.0000
- Fold 4 Accuracy: 1.0000

The consistent and perfect accuracy of 1.0000 across all folds highlights the exceptional performance of the AdaBoost classifier. This implies the model's proficiency in accurately classifying instances within different subsets of the financial dataset.

The ensemble approach, where weak learners are combined to form a strong model, contributes to AdaBoost's robustness and generalization ability. The model's ability to achieve

perfect accuracy across various folds underscores its effectiveness in capturing complex patterns within the financial data, making it a compelling choice for tasks demanding high precision.

In conclusion, the AdaBoost classifier, with its ensemble learning strategy and the selected parameter configuration, demonstrates outstanding performance in binary classification for financial datasets. Its consistent and perfect accuracy underscores its proficiency and generalization ability, making it a powerful tool for accurate prediction in the dynamic and nuanced financial domain.

Logistic Regression

In the quest for developing an effective binary classification model tailored to a financial dataset, the Logistic Regression algorithm, a widely-used linear classification method, is chosen. Logistic Regression is particularly effective for binary classification tasks, providing a straightforward yet powerful approach. The key characteristic of Logistic Regression is its ability to model the probability of an instance belonging to a certain class.
Algorithm and Process:
The Logistic Regression model is meticulously trained and evaluated using a KFold cross-validation framework. In each fold, the financial dataset is systematically divided into training and validation sets. The Logistic Regression model is then trained on the training set, optimizing its parameters to maximize the likelihood of correctly classifying instances. Following the training phase, the model's performance is evaluated on the validation set, with accuracy serving as the chosen evaluation metric. Accuracy represents the proportion of correctly classified instances, providing a straightforward measure of the model's predictive capability.
Accuracy Results:

- - Fold 0 Accuracy: 0.8073
- - Fold 1 Accuracy: 0.8073
- - Fold 2 Accuracy: 0.8073
- - Fold 3 Accuracy: 0.8073
- - Fold 4 Accuracy: 0.8073

The Logistic Regression classifier consistently achieves an accuracy of approximately 0.8073 across all folds. While Logistic Regression may be considered a simpler model compared to some ensemble methods, its ability to consistently provide a reasonable level of accuracy for binary classification tasks is noteworthy.
The uniformity in accuracy values across folds suggests a stable performance of the Logistic Regression model. This consistency indicates the model's capability to generalize well to unseen data, a crucial aspect in ensuring its reliability when applied to new instances.
In conclusion, the Logistic Regression classifier, with its simplicity and reliability, demonstrates a consistent and stable performance in binary classification for the financial dataset. While its accuracy may not reach the levels of more complex models, Logistic Regression remains a valuable tool, especially when interpretability and ease of implementation are important considerations in the financial domain.

Neural Network

For the training and evaluation of data, a Multi-Layer Perceptron (MLP) classifier using a KFold cross-validation approach is used. The for loop iterates through each fold of the KFold cross-validation, splitting the training data into training and validation sets. Within each fold, the training data is further split using the train_test_split function into training and validation sets (X_train_imputed_fold, X_valid_imputed_fold, y_train_fold, y_valid_fold). An MLP classifier model is created using MLPClassifier with a single hidden layer of 100 neurons and a maximum of 100 iterations. The model is trained on the training data (X_train_imputed_fold, y_train_fold) using the fit method. Predictions are made on the validation set (X_valid_imputed_fold), and the results are stored in the predictions variable. The accuracy of the model is calculated using the accuracy_score function, comparing the predicted labels with the true labels from the validation set.The fold index, start time, and accuracy for each fold are printed to the console.
The output indicates that, for each fold, the MLP classifier achieved an accuracy of 62.50%.

Conclusion

The selection of the best model for a financial application is a nuanced decision that depends on various factors, including the specific objectives and requirements of the task at hand. In instances where precision is of paramount importance, XGBoost and AdaBoost emerge as strong contenders due to their ability to achieve high accuracy.
XGBoost, an optimized gradient boosting algorithm, is particularly well-suited for handling complex relationships within financial datasets. Its ensemble approach, combining the strengths of multiple weak learners, allows it to capture intricate patterns and make accurate predictions. Moreover, XGBoost offers regularization techniques, which help prevent overfitting and enhance its generalization capabilities. This is crucial in financial applications where robust performance on unseen data is vital.
Similarly, AdaBoost, with its focus on correcting the errors of previous models in the ensemble, can be effective in scenarios where precision is a critical metric. AdaBoost iteratively emphasizes misclassified instances, leading to a strong overall model that excels in minimizing errors.
However, it's essential to acknowledge that the choice of a complex model like XGBoost or AdaBoost may come with increased computational demands and complexity in hyperparameter tuning. These considerations need to be weighed against the potential gains in accuracy.
On the other hand, Logistic Regression, while simpler compared to boosting algorithms, offers a stable and interpretable solution. In financial applications, interpretability is often a crucial factor, especially when decisions impact real-world financial transactions or compliance. Logistic Regression provides a clear understanding of the impact of each feature on the prediction, making it easier for stakeholders to comprehend and trust the model.
The trade-off between model complexity and interpretability should be carefully evaluated based on the specific requirements of the financial application. In situations where a balance between accuracy and interpretability is sought, Logistic Regression might be the preferred choice. Its

simplicity not only aids in better understanding but can also lead to faster deployment and easier maintenance.

In conclusion, the optimal choice of a model for financial applications hinges on a careful consideration of the trade-offs between accuracy, interpretability, and computational complexity. While XGBoost and AdaBoost excel in precision and capturing complex patterns, Logistic Regression offers a stable and interpretable alternative. The decision ultimately rests on the unique needs and constraints of the financial domain in question.